



UNIVERSITY INSTITUTE OF
COMPUTING

PROJECT REPORT

ON

GPA & CGPA CALCULATOR

Name : Ruchi Taneja
UID : 24BCA20019
Section : 24BCV-1A
Subject : OOPS
Subject Code : 24CAH20

Submitted to : Mr. Jitendra
Designation : Assistant Professor
Signature : _____

1. Introduction

The **GPA & CGPA Calculator** is a simple console-based C++ application developed to demonstrate the practical use of **Object-Oriented Programming (OOP)** concepts such as modularity and abstraction.

The program helps students calculate their **Grade Point Average (GPA)** for a semester and **Cumulative Grade Point Average (CGPA)** over multiple semesters. It operates without any database, only using functions, arrays, and console input/output for interaction.

This project was designed as a small yet efficient academic utility that allows users to perform GPA and CGPA computations quickly, through a clear and menu-driven interface.

2. Objectives of the Project

1. To demonstrate the use of modular programming through user-defined functions.
2. To provide an easy-to-use tool for calculating GPA and CGPA interactively.
3. To apply OOP principles like **abstraction** and **function modularity**.
4. To make use of loops, arrays, and switch statements effectively in real-world logic.
5. To encourage error-handling and smooth user navigation through menus.

3. Scope of the Project

This project is designed for **students and academic environments**, where quick grade computation is needed.

It manages the following operations:

- GPA calculation for a given semester.
- CGPA calculation across multiple semesters.
- Displaying the mathematical method used for GPA and CGPA computation.
- Providing menu-driven control for easy navigation and re-calculation.

All operations are done **in-memory** during runtime, no data storage or external libraries are used.

4. System Design

Modules –

1. GPA Module

- Accepts subject-wise credits and grade points.
- Calculates the semester GPA using the formula:
$$\text{GPA} = (\text{Sum of (Credit} \times \text{Point)}) / \text{Total Credits}.$$

2. CGPA Module

- Takes multiple semester GPAs as input.
- Computes the cumulative GPA using:
$$\text{CGPA} = (\text{Sum of GPAs}) / \text{Number of Semesters}.$$

3. Method Module

- Displays the formulas and concepts behind GPA and CGPA calculation.

4. Main Menu Module

- Controls program flow.
- Provides options to calculate GPA, CGPA, view the method, or exit the program.

5. Source Code

```
#include <iostream>
```

```
#include <stdlib.h>
```

```
using namespace std;
```

```
void calculateGPA();
```

```
void calculateCGPA();
```

```
void method();
```

```
int main()
```

```
{
```

```
    system("cls");
```

```
    int input;
```

```
    cout<<"-----"<<endl;
```

```
    cout<<"          GPA & CGPA Calculator          "<<endl;
```

```
    cout<<"-----\n"<<endl;
```

```
    cout<<"    MENU:"<<endl;
```

```
    cout<<"        1. Calculate GPA (Grade Point Average)"<<endl;
```

```
    cout<<"        2. Calculate CGPA (Cummulative Grade Point Average)"<<endl;
```

```
    cout<<"        3. Method that is applied here for calclating GPA & CGPA"<<endl;
```

```
    cout<<"        4. Exit Application"<<endl;
```

```
cout<<"-----"<<endl;
```

```
sub:
```

```
cout<<"Enter your choice: ";
```

```
cin>>input;
```

```
switch(input)
```

```
{
```

```
case 1:
```

```
    calculateGPA();
```

```
    break;
```

```
case 2:
```

```
    calculateCGPA();
```

```
    break;
```

```
case 3:
```

```
    method();
```

```
    break;
```

```
case 4:
```

```
    exit(EXIT_SUCCESS);
```

```
    break;
```

```
default:
```

```
    cout<<"You have entered wrong input. Try again!\n"<<endl;
```

```
    goto sub;
```

```
    break;
```

```
}
```

```
}
```

```
void calculateGPA()
```

```
{
```

```
int q;
```

```
system("cls");
```

```
cout<<"----- GPA Calculating -----"<<endl;
```

```
cout<<" How many subject's points do you want to calculate? : ";
```

```
cin>>q;
```

```
float credit[q];
```

```
float point[q];
```

```
for(int i=0;i<q;i++)
```

```
{
```

```
    cout<<"Enter the credit for the subject "<<i+1<<": ";
```

```
    cin>>credit[i];
```

```
    cout<<"Enter the point of the subject "<<i+1<<": ";
```

```
    cin>>point[i];
```

```
    cout<<"-----\n\n";
```

```
}
```

```
float sum=0, totCr=0;
```

```
for(int j=0;j<q;j++)
```

```
{
```

```
    sum += credit[j]*point[j];
```

```
    totCr += credit[j];
```

```
}
```

```
cout<<"\nTotal Points: "<<sum<<" | Total Credits: "<<totCr<<" | Total GPA: "<<sum/totCr<<"\n";
```

```
sub:
```

```
int inmenu;
```

```
cout<<"\n1. Calculate Again\n2. Go Back to Main Menu\n3. Exit This App\nYour Input: ";
```

```
cin>>inmenu;
```

```
switch(inmenu)
```

```
{
```

```
    case 1: calculateGPA(); break;
```

```

        case 2: main(); break;
        case 3: exit(EXIT_SUCCESS);
        default: cout<<"Wrong Input! Try again!"; goto sub;
    }
}

```

```

void calculateCGPA()
{
    system("cls");
    int l;
    cout<<"----- CGPA Calculating -----\\n";
    cout<<"How many semester results do you want input? :";
    cin>>l;

```

```

    float semrs[l];
    for(int i=0;i<l;i++)
    {
        cout<<"Enter Semester "<<i+1<<" Result (GPA): ";
        cin>>semrs[i];
    }

```

```

    float semtot=0;
    for(int j=0;j<l;j++)
        semtot+=semrs[j];

```

```

    cout<<"\\n***** Your CGPA is "<<semtot/l<<" *****\\n";

```

sub:

```

    int inmenu;
    cout<<"\\n1. Calculate Again\\n2. Go Back to Main Menu\\n3. Exit This App\\nYour Input: ";
    cin>>inmenu;

```

```

switch(inmenu)
{
    case 1: calculateCGPA(); break;
    case 2: main(); break;
    case 3: exit(EXIT_SUCCESS);
    default: cout<<"Wrong Input! Try again!"; goto sub;
}
}

```

```

void method()
{
    system("cls");
    cout<<"----- Method of Calculating GPA & CGPA -----\\n";
    cout<<"GPA = (Sum of (Credit × Point)) / (Sum of Credits)\\n";
    cout<<"CGPA = (Sum of GPAs) / (Number of Semesters)\\n";
    cout<<"-----\\n";
}

```

```

sub:
    int inmenu;
    cout<<"1. Go Back to Main Menu\\n2. Exit This App\\nYour Input: ";
    cin>>inmenu;

```

```

switch(inmenu)
{
    case 1: main(); break;
    case 2: exit(EXIT_SUCCESS);
    default: cout<<"Wrong Input! Try again!"; goto sub;
}
}

```

6. Sample Output

```
-----  
GPA & CGPA Calculator  
-----  
  
MENU:  
1. Calculate GPA (Grade Point Average)  
2. Calculate CGPA (Cumulative Grade Point Average)  
3. Method of Calculation  
4. Exit Application  
-----  
  
Enter your choice: 1  
----- GPA Calculating -----  
  
How many subjects? : 3  
Enter credit for subject 1: 3  
Enter point for subject 1: 9  
Enter credit for subject 2: 4  
Enter point for subject 2: 8  
Enter credit for subject 3: 2  
Enter point for subject 3: 10  
  
Total Points: 77.0 | Total Credits: 9.0 | Total GPA: 8.55
```

7. Advantages

1. Simple, fast, and beginner-friendly.
2. No external dependencies or databases required.
3. Clear menu-driven interface.
4. Demonstrates modular programming and abstraction effectively.
5. Easily extendable for future academic tools.

8. Limitations

1. Data is not saved after program exit.
2. No error validation for invalid numerical inputs.
3. Limited visual design (console-based only).
4. Does not support weighted GPAs across different grading systems.

9. Future Enhancements

1. Add file handling to save previous GPA and CGPA data.
2. Include subject and semester history reports.
3. Introduce graphical or GUI interface for better visualization.
4. Integrate grading systems (like letter-to-point conversion).
5. Add login functionality for multi-user access.

10. Conclusion

This project successfully demonstrates how **modular programming** and **basic OOP concepts** can be used to build a real-world calculator application in C++.

Through functions, loops, and user interaction, the program provides a reliable way to compute GPA and CGPA efficiently. It enhances understanding of:

- User-defined functions
- Menu-driven design
- Mathematical computation using arrays

Hence, the **GPA & CGPA Calculator** meets its objectives of applying C++ fundamentals to solve a practical academic problem effectively.